Name: **Rahul Nalawade** Roll No: **201351017**

Indian Institute of Information Technology, Vadodara

CS308 – Introduction to Artificial Intelligence

Assignment #02

1. Implement Linear classification with analytical solution.

#Source code:

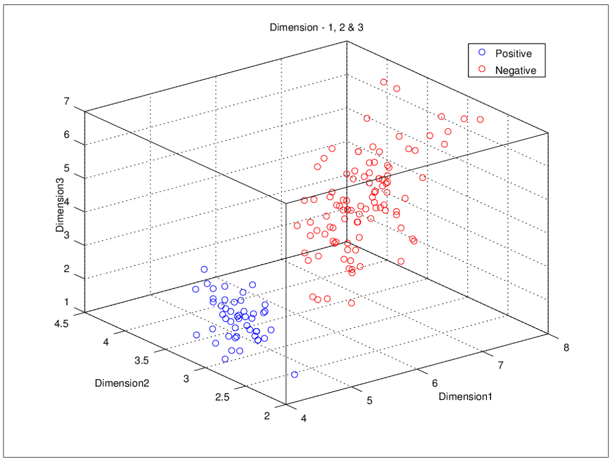
```
clear all;
close all;

m_train = load("-ascii", "iris_data_norm_train.txt");

m_test = load("-ascii", "iris_data_norm_test.txt");

[w, no_of_iterations, Ein] = training_pla(m_train);

printf("-------------------------------------------------------\n");
printf("Machine has completed learning,\n");
printf("from the given dataset.\n");
printf("So, weights obtained - \n");

w

[misclassifications, classifications] = testing_pla(m_test,w);
printf("-------------------------------------------------------\n");

misclassifications
classifications

printf("Accuracy = %f \n", (100*classifications)/ (misclassifications +
classifications));
printf("-------------------------------------------------------\n");

m_both = load("-ascii", "iris_data_norm_both.txt");
plotting_datapoints(m_both);
```

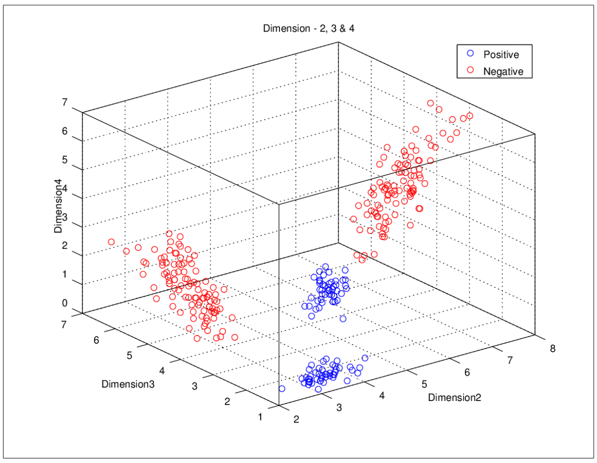
Name: **Rahul Nalawade** Roll No: **201351017**

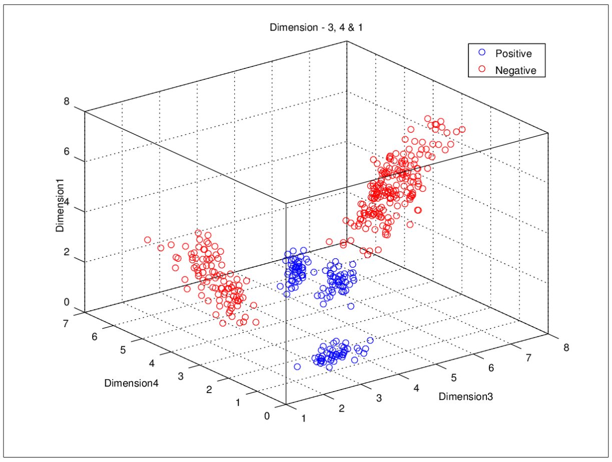
#Output:

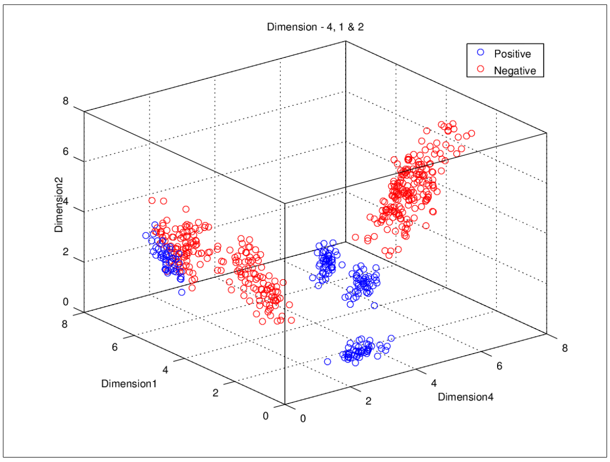
```
octave:1> lab21
-----------------------------------------------------
Machine has completed learning, from the given training dataset.
So, weights obtained -
W =

   1.8173   6.9173  -7.9827  -2.9827

------------------------------------------------
misclassifications = 0
classifications =  40
Accuracy = 100.000000
-----------------------------------------------------
```

#Diagrams:









2. Shuffle the training data (using a common seed, 1234) and then prepare train-test partitions according to (30/70, 40/60, 50/50, 60/40 and 70/30) percent sizes. Train on train partition and test on both. Report E_in and E_out for each partition.

#Source Code:

```
clear all;
close all;

m_both = load("-ascii", "iris_data_norm_both.txt");

[L, W] = size(m_both);

p = input("Enter the training percentage: ");

N = (p/100)*L;

m_train = m_both(1:N,1:W);

m_test = m_both(N+1:L,1:W);

printf("-------------------------------------------------------\n");
printf("Machine has completed learning,\n");
printf("from the %d%% of the given dataset.\n");
printf("So, weights obtained - \n",p);

[weights, no_of_iterations, Ein] = training_pla_it(m_train)

w=zeros(1,W-1);

for i=1:no_of_iterations
      w = weights(i, 1:(W-1));
      [misclassifications, classifications] = testing_pla(m_test,w);
      Eout(i) = misclassifications;
end

Eout
```

Name: **Rahul Nalawade** Roll No: **201351017**

#Output:

```
octave:1> lab22
Enter the training percentage: 30
-------------------------------------------------------
Machine has completed learning,
from the 30% of the given dataset.
So, weights obtained -
weights =

   0.42592   5.22592  -9.07408  -3.27408
   0.42592   5.22592  -9.07408  -3.27408

no_of_iterations =  2
Ein =

   7   0

Eout =

   0   0
```

```
octave:2> lab22
Enter the training percentage: 40
-------------------------------------------------------
Machine has completed learning,
from the 40% of the given dataset.
So, weights obtained -
weights =

   0.42426   5.22426  -9.07574  -3.27574
   0.42426   5.22426  -9.07574  -3.27574

no_of_iterations =  2
Ein =

   7   0

Eout =

   0   0
```

```
octave:3> lab22
Enter the training percentage: 50
-------------------------------------------------------
Machine has completed learning,
from the 50% of the given dataset.
So, weights obtained -
weights =

   1.5616   5.9616  -7.3384  -2.7384
   1.5616   5.9616  -7.3384  -2.7384

no_of_iterations =  2
Ein =

   7   0

Eout =

   0   0
```

```
octave:4> lab22
Enter the training percentage: 60
-------------------------------------------------------
Machine has completed learning,
from the 60% of the given dataset.
So, weights obtained -
weights =

   0.79615   5.59615  -8.70385  -2.90385
   0.79615   5.59615  -8.70385  -2.90385

no_of_iterations =  2
Ein =

   7   0

Eout =

   0   0
```

```
octave:5> lab22
Enter the training percentage: 70
-------------------------------------------------------
Machine has completed learning,
from the 70% of the given dataset.
So, weights obtained -
weights =

   0.43610   5.23610  -9.06390  -3.26390
   0.43610   5.23610  -9.06390  -3.26390

no_of_iterations =  2
Ein =

   7   0

Eout =

   0   0
```

Name: **Rahul Nalawade** Roll No: **201351017**

3. For PLA, plot learning curves. Where X axis will be number of iterations and Y axis will be normalized error (misclassifications) on train (E_in) and test E_out). Do it for each partitions prepared in task 2.

#Source Code:

```
clear all;
close all;

m_both = load("-ascii", "iris_data_norm_both.txt");

[L, W] = size(m_both);

p = input("Enter the training percentage: ");

N = (p/100)*L;

m_train = m_both(1:N,1:W);

m_test = m_both(N+1:L,1:W);

printf("-------------------------------------------------------\n");
printf("Machine has completed learning,\n");
printf("from the %d%% of the given dataset.\n");
printf("So, weights obtained - \n",p);

[weights, no_of_iterations, Ein] = training_pla_it(m_train);

w = zeros(1,W-1);

for i=1:no_of_iterations
      w = weights(i, 1:(W-1));
      [misclassifications, classifications] = testing_pla(m_test,w);
      Eout(i) = misclassifications;
end

Eout;

weights

Einp = Ein.*(100/N)

Eoutp = Eout.*(100/(L-N))
```

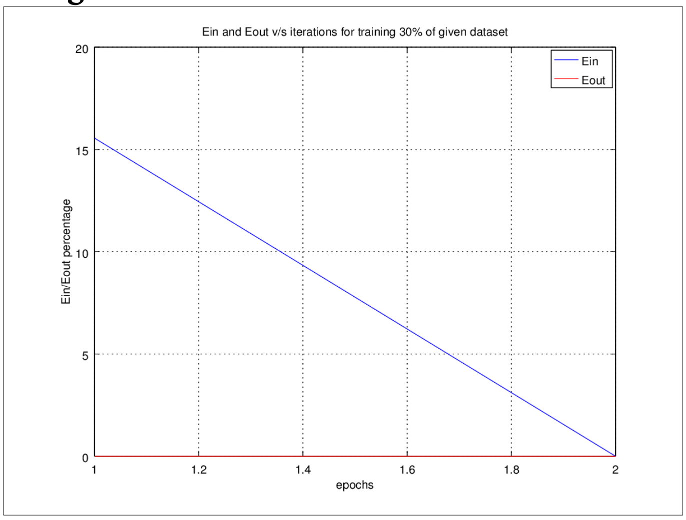
Name: **Rahul Nalawade** Roll No: **201351017**

```
% plotting the graph
plot(1:no_of_iterations, Einp,'b', 1:no_of_iterations, Eoutp,'r');
grid on;
hold on;
title(strvcat(["Ein and Eout v/s iterations for training ",int2str(p),"% of given dataset"]));
xlabel('epochs');
ylabel('Ein/Eout percentage');
legend('Ein', 'Eout');
print(strvcat(["Ein and Eout ",int2str(p),"%% dataset.png"]), '-dpng');
```

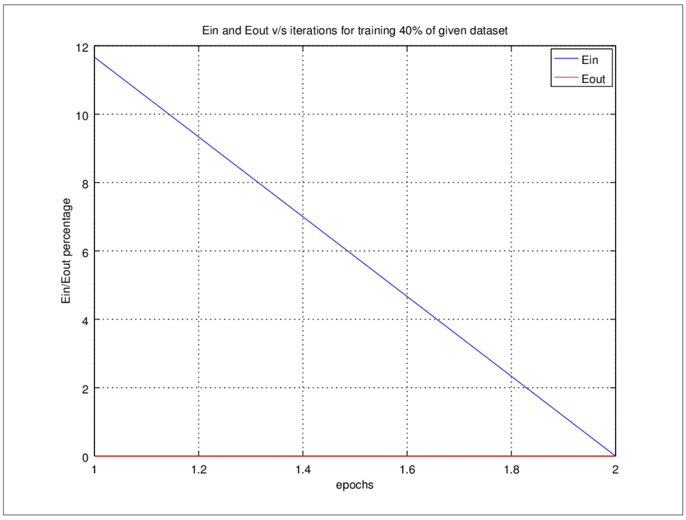
#Output:

```
octave:6> lab23
Enter the training percentage: 30
-----------------------------------------------------
Machine has completed learning,
from the 30% of the given dataset.
weights =

   1.2970   6.0970  -8.2030  -2.4030
   1.2970   6.0970  -8.2030  -2.4030

Einp =

   15.55556    0.00000

Eoutp =

   0   0
```

#Diagrams:

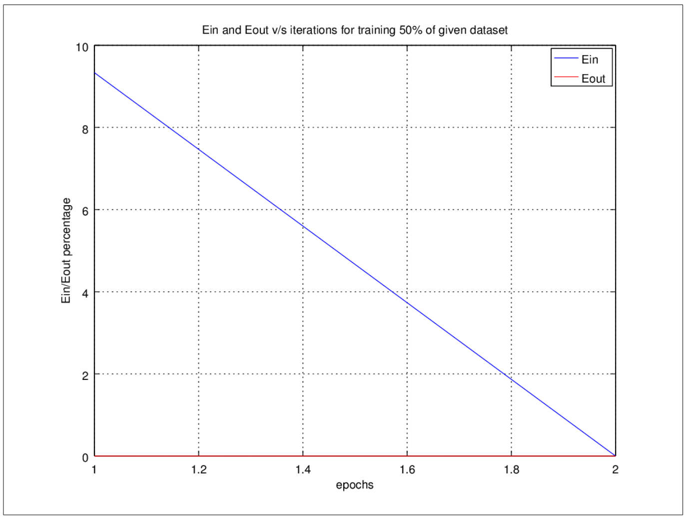



```
octave:7> lab23
Enter the training percentage: 40
-----------------------------------------------------
Machine has completed learning,
from the 40% of the given dataset.
weights =

   0.48568   5.28568  -9.01432  -3.21432
   0.48568   5.28568  -9.01432  -3.21432

Einp =

   11.66667    0.00000

Eoutp =

   0   0
```

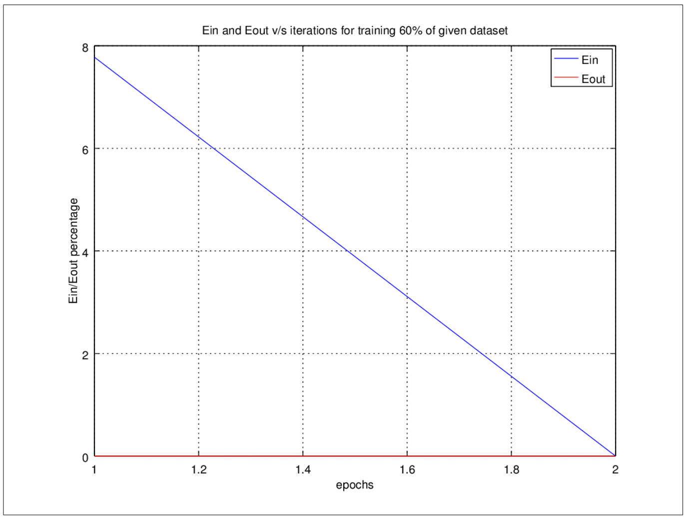


Name: **Rahul Nalawade** Roll No: **201351017**

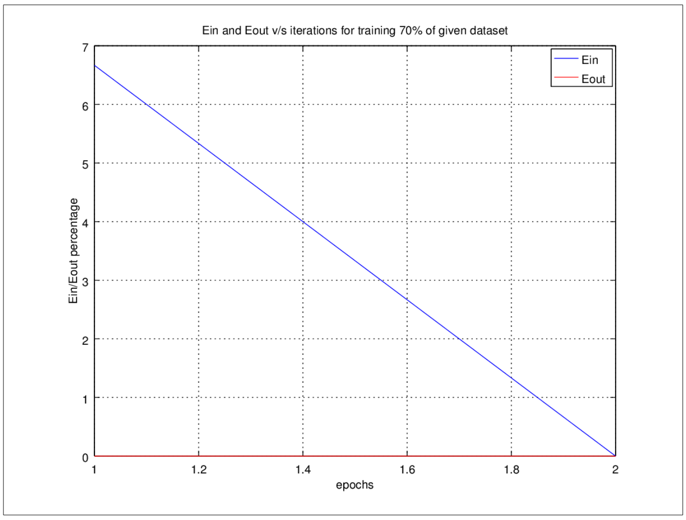
```
Enter the training percentage: 50
-------------------------------------------------------
Machine has completed learning,
from the 50% of the given dataset.
weights =

   1.1452   5.9452  -8.3548  -2.5548
   1.1452   5.9452  -8.3548  -2.5548

Einp =

   9.33333   0.00000

Eoutp =

   0   0
```




```
Enter the training percentage: 60
-------------------------------------------------------
Machine has completed learning,
from the 60% of the given dataset.
weights =

   0.60404   5.40404  -8.89596  -3.09596
   0.60404   5.40404  -8.89596  -3.09596

Einp =

   7.77778   0.00000

Eoutp =

   0   0
```




```
octave:10> lab23
Enter the training percentage: 70
-------------------------------------------------------
Machine has completed learning,
from the 70% of the given dataset.
weights =

   0.66348   5.46348  -8.83652  -3.03652
   0.66348   5.46348  -8.83652  -3.03652

Einp =

   6.66667   0.00000

Eoutp =

   0   0
```



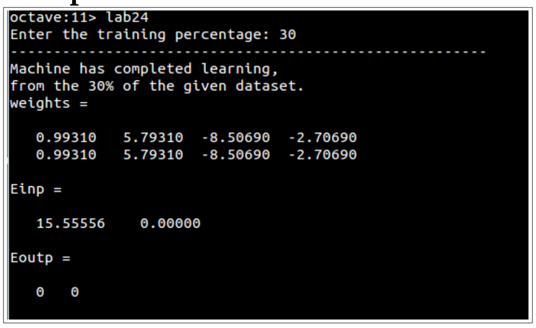
4. Implement Pocket algorithm for PLA. And replot task3 .

#Source Code:

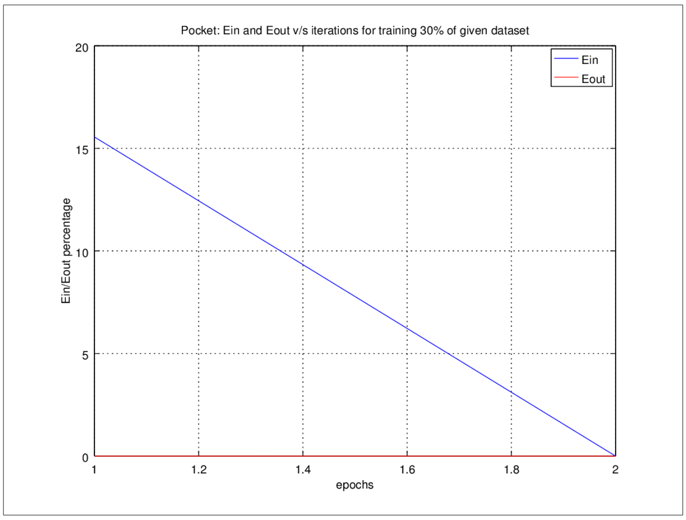
```
clear all;
close all;

m_both = load("-ascii", "iris_data_norm_both.txt");

[L, W] = size(m_both);

p = input("Enter the training percentage: ");

N = (p/100)*L;

m_train = m_both(1:N,1:W);

m_test = m_both(N+1:L,1:W);

printf("-------------------------------------------------------\n");
printf("Machine has completed learning,\n");
printf("from the %d%% of the given dataset.\n");
printf("So, weights obtained - \n",p);

[weights, no_of_iterations, Ein] = training_pocket(m_train);

w=zeros(1,W-1);

for i=1:no_of_iterations
      w = weights(i, 1:(W-1));
      [misclassifications, classifications] = testing_pla(m_test,w);
      Eout(i) = misclassifications;
end

Eout;

weights

Einp = Ein.*(100/N)

Eoutp = Eout.*(100/(L-N))


% plotting the graph
plot(1:no_of_iterations, Einp,'b', 1:no_of_iterations, Eoutp,'r');
```

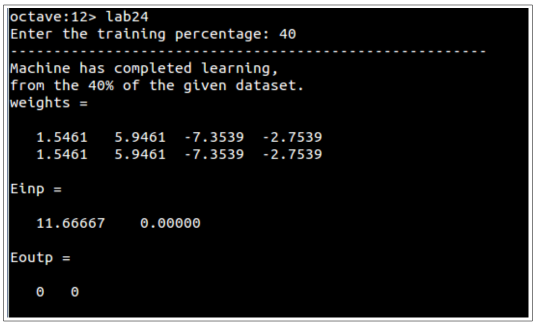
```
grid on;
hold on;
title(strvcat(["Pocket: Ein and Eout v/s iterations for training
",int2str(p),"% of given dataset"]));
xlabel('epochs');
ylabel('Ein/Eout percentage');
legend('Ein', 'Eout');
print(strvcat(["Pocket: Ein and Eout ",int2str(p),"%% dataset.png"]), '-
dpng');
```

#Output:

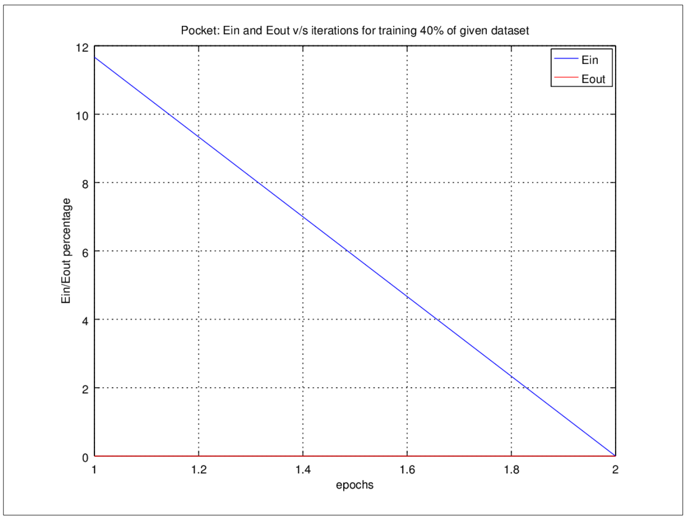


```
octave:11> lab24
Enter the training percentage: 30
------------------------------------------------------
Machine has completed learning,
from the 30% of the given dataset.
weights =

   0.99310   5.79310  -8.50690  -2.70690
   0.99310   5.79310  -8.50690  -2.70690

Einp =

   15.55556    0.00000

Eoutp =

   0   0
```



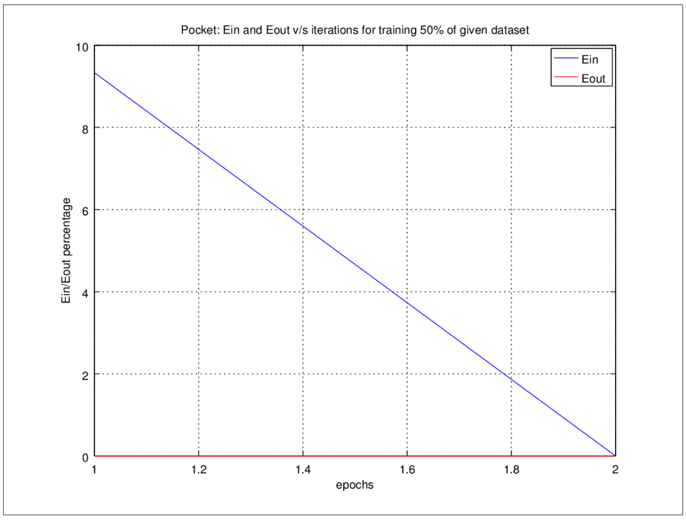


```
octave:12> lab24
Enter the training percentage: 40
------------------------------------------------------
Machine has completed learning,
from the 40% of the given dataset.
weights =

   1.5461   5.9461  -7.3539  -2.7539
   1.5461   5.9461  -7.3539  -2.7539

Einp =

   11.66667    0.00000

Eoutp =

   0   0
```

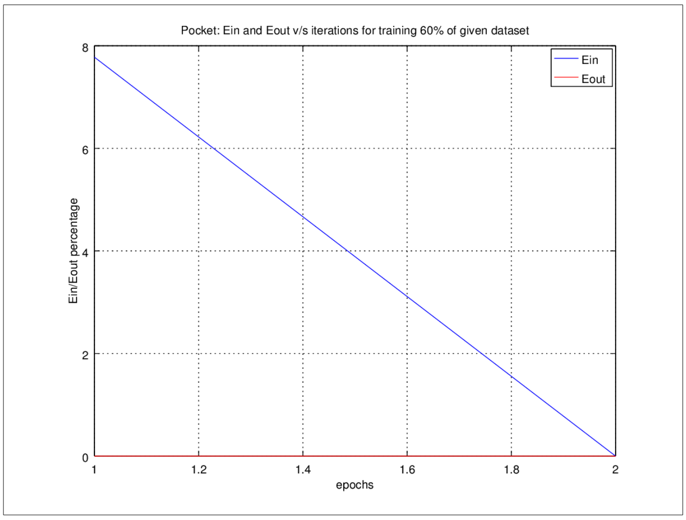


Name: **Rahul Nalawade** Roll No: **201351017**

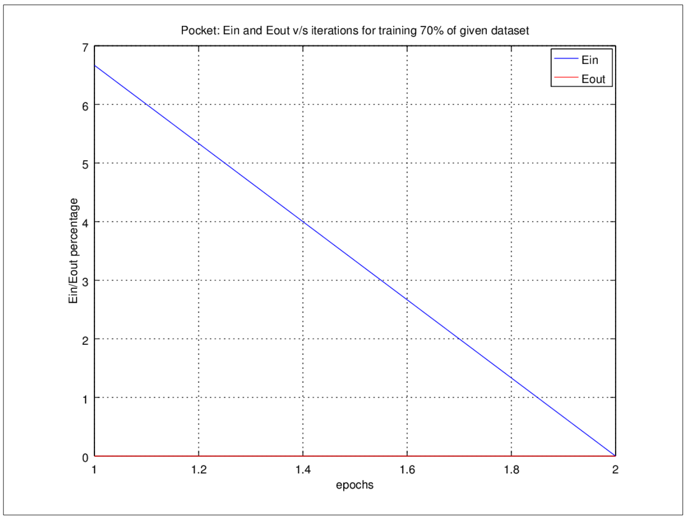
```
octave:13> lab24
Enter the training percentage: 50
-------------------------------------------------------
Machine has completed learning,
from the 50% of the given dataset.
weights =

   1.5378   5.9378  -7.3622  -2.7622
   1.5378   5.9378  -7.3622  -2.7622

Einp =

   9.33333   0.00000

Eoutp =

   0   0
```




```
octave:14> lab24
Enter the training percentage: 60
-------------------------------------------------------
Machine has completed learning,
from the 60% of the given dataset.
weights =

   1.2787   6.0787  -8.2213  -2.4213
   1.2787   6.0787  -8.2213  -2.4213

Einp =

   7.77778   0.00000

Eoutp =

   0   0
```




```
Enter the training percentage: 70
-------------------------------------------------------
Machine has completed learning,
from the 70% of the given dataset.
weights =

   0.98135   5.78135  -8.51865  -2.71865
   0.98135   5.78135  -8.51865  -2.71865

Einp =

   6.66667   0.00000

Eoutp =

   0   0
```



5. Implement the linear regression model on train dataset and test the result (w) on the test dataset. {as discussed during the lab hours}

#Source Code:

```
clear all;
close all;


m_train = load("-ascii", "iris_data_norm_train.txt");


w = training_linreg(m_train)

m_test = load("-ascii","iris_data_norm_test.txt");

printf("-----------------------------------------------------\n");
[misclassifications, classifications] = testing_pla(m_test, w)

printf("Accuracy = %f \n", (100*classifications)/(misclassifications +
classifications));
printf("-----------------------------------------------------\n");
```

#Output:

```
octave:16> lab25
w =

   0.025827   0.406082  -0.428349  -0.089038

----------------------------------------------------
misclassifications = 0
classifications =  40
Accuracy = 100.000000
---------------------------------------------------------
```

6. Implement the II order non-linear regression model for train dataset and output the results. {as discussed in the lab hours}

#Source Code:

```
clear all;
close all;


m_train = load("-ascii", "iris_data_norm_train.txt");

printf("The weight vector for II order non-linear regression:\n");

w = training_nonlin(m_train)
```

#Output:

```
octave:17> lab26
The weight vector for II order non-linear regression:
w =

  -4.556317
   2.087750
   1.123430
  -1.927859
   0.116359
  -0.187289
  -0.025208
   0.138921
   0.497422
  -0.141351
   0.148946
  -0.296759
   0.093756
  -0.399300
   0.116409

--------------------------------------------------
```

#Additional Functions Codes:

```
function plotting_datapoints(M)

% plots the data points taking 3 dimensions at a time
% M = [attribute1 attribute2 . . . attributeN targetFunction];
% produces all the 3 at a time 3D plots
% as Dimension - 1, 2 & 3.png (say, for first three dimensions)

[len, A] = size(M);
A--;

X = [M(1:len,1:A)];
T = [M(1:len,A+1)];

p=0;
n=0;

for i=1:len

      if(T(i) == 1)
            Positive(++p) = i;
      else
            Negative(++n) = i;
      end
end
```

```
for j=1:A

      switch j

      case A-1,
            d1 = j;
            d2 = j+1;
            d3 = 1;

      case A,
            d1 = j;
            d2 = 1;
            d3 = 2;

      otherwise,
            d1 = j;
            d2 = j+1;
            d3 = j+2;
      end

      s0 = strvcat(["Dimension - ",int2str(d1),", ",int2str(d2)," &
",int2str(d3)]);

      s1 = strvcat(["Dimension",int2str(d1)]);
      s2 = strvcat(["Dimension",int2str(d2)]);
      s3 = strvcat(["Dimension",int2str(d3)]);

      for k=1:length(Positive)

            xp(k) = X((Positive(k)),d1);
            yp(k) = X((Positive(k)),d2);
            zp(k) = X((Positive(k)),d3);
      end

      for k=1:length(Negative)

            xn(k) = X((Negative(k)),d1);
            yn(k) = X((Negative(k)),d2);
            zn(k) = X((Negative(k)),d3);
      end

      plot3(xp,yp,zp,'bo',xn,yn,zn,'ro')
      hold on
```

```
      grid on
      xlabel(s1);
      ylabel(s2);
      zlabel(s3);
      legend('Positive','Negative')
      title(s0);
      print(strvcat([ s0,".png"]),'-dpng');

end
```

```
function [W,ITERATIONS,E] = training_pla(M)

% trainig_pla(matrix) trains on the data given in the form of matrix
% M = [attribute1 attribute2 . . . attributeN targetFunction];
% produces hypothesis weights w = {w1, w2, . . ., wN}

[len, A] = size(M);
A--;

X = M(1:len,1:A);
T = M(1:len,A+1);

w(1:A) = rand();

iterations = 0;
improvements = 0;
i = 0;

while(i < len)

      i++;

      x = transpose(X(i,1:A));

      product = w * x;

      if((product < 0 && T(i) == 1) || (product > 0 && T(i) == -1))
            improvements++;
            w = w + T(i)* x';      % w <- w + yx
      end

      if (i == len && improvements == 0)      % last iteration
            iterations++;
```

```
		errors(iterations) = improvements;
	end

	if(i == len && improvements > 0)
		i = 0;
		iterations++;
		errors(iterations) = improvements;
		improvements = 0;		% restarting the iterations
	end

end
W = w;
ITERATIONS = iterations;
E = errors;
endfunction
```

```
function [E,C] = testing_pla(M,w)

% testing_pla(matrix) tests on the data given in the form of matrix
% M = [attribute1 attribute2 . . . attributeN targetFunction];
% w = [w1 w2 . . . wN];
% produce E, percentage misclassifications

[len, A] = size(M);
A--;

X = [M(1:len,1:A)];
T = [M(1:len,A+1)];

err = 0;
corr = 0;

for i = 1:len

	x = transpose([X(i,1:A)]);

	product = w * x;

	if((product < 0 && T(i) == 1) || (product > 0 && T(i) == -1))
		err++;
	else
		corr++;
	end
```

```
end

E = err;
C = corr;
endfunction
```

```
function [W,ITERATIONS,E] = training_pla_it(M)

% trainig_pla_it(matrix) trains on the data given in the form of matrix
ITERATIVELY
% M = [attribute1 attribute2 . . . attributeN targetFunction];
% produces hypothesis weights w = {w1, w2, . . ., wN} for each ITERATION

[len, A] = size(M);
A--;

X = M(1:len,1:A);
T = M(1:len,A+1);

w(1:A) = rand();

iterations = 0;
improvements = 0;
i = 0;

while(i < len)

      i++;

      x = transpose(X(i,1:A));

      product = w * x;

      if((product < 0 && T(i) == 1) || (product > 0 && T(i) == -1))
            improvements++;
            w = w + T(i)* x';      % w <- w + yx
      end

      if (i == len && improvements == 0)      % last iteration
            iterations++;
            errors(iterations) = improvements;
            w_it(iterations,1:A) = w;
      end
```

```
      if(i == len && improvements > 0)
            i = 0;
            iterations++;
            errors(iterations) = improvements;
            w_it(iterations,1:A) = w;
            improvements = 0;            % restarting the iterations
      end

end

W = w_it;
ITERATIONS = iterations;
E = errors;
endfunction
```

```
function [W,ITERATIONS,E] = training_pocket(M)

% trainig_pocket(matrix) trains on the data given in the form of matrix
% using POCKET ALGORITHM
% M = [attribute1 attribute2 . . . attributeN targetFunction];

[len, A] = size(M);
A--;

X = M(1:len,1:A);
T = M(1:len,A+1);

w(1:A) = rand();

iterations = 0;
improvements = 0;
i = 0;

while(i < len)

      i++;

      x = transpose(X(i,1:A));

      product = w * x;

      if((product < 0 && T(i) == 1) || (product > 0 && T(i) == -1))
```

```
            improvements++;
            w = w + T(i)* x';      % w <- w + yx
      end

      if (i == len && improvements == 0)     % last iteration
            iterations++;
            errors(iterations) = improvements;

            if(iterations == 1)
                  w_it(iterations,1:A) = w;
                  errors(iterations) = improvements;

            elseif(iterations > 1 && errors(iterations-1) <
errors(iterations) )   % Condition of POCKET ALGORITHM
                  w_it(iterations,1:A) = w_it(iterations - 1,1:A);
                  errors(iterations) = errors(iterations - 1);
            else
                  w_it(iterations,1:A) = w;
                  errors(iterations) = improvements;
            end
      end

      if(i == len && improvements > 0)
            iterations++;
            errors(iterations) = improvements;

            if(iterations >= 2 && errors(iterations - 1) < errors(iterations)
)           % Condition of POCKET ALGORITHM
                  w_it(iterations,1:A) = w_it(iterations - 1,1:A);
                  errors(iterations) = errors(iterations - 1);
            else
                  w_it(iterations,1:A) = w;
                  errors(iterations) = improvements;
            end
            i = 0;
            improvements = 0;           % restarting the iterations
      end

      if(iterations == 1000)
            break;
      end

end
```

```
W = w_it;
ITERATIONS = iterations;
E = errors;
endfunction
```

```
function W = training_linreg(M)

% trainig_linreg(matrix) trains on the data given in the form of matrix
% using LINEAR REGRESSION
% M = [attribute1 attribute2 . . . attributeN targetFunction];
% produces hypothesis weights w = {w1, w2, . . ., wN}

[len, A] = size(M);
A--;

X = M(1:len,1:A);
T = M(1:len,A+1);

w = pinv(X)*T;
W = w';
```

```
function W = training_nonlin(M)

% trainig_momlin(matrix) trains on the data given in the form of matrix
% using NON LINEAR REGRESSION for degree = 2
% M = [attribute1 attribute2 . . . attributeN targetFunction];
% produces hypothesis weights w = {w1, w2, . . ., wN}

[len, A] = size(M);
A--;

Max = A*(A+1)/2;

x(1:len,1) = ones(len,1);

for i=2:A+1
      x(1:len,i) = M(1:len,i-1);
end

for i=6:9
      x(1:len,i) = M(1:len,(i-5)) .* M(1:len,(i-5));
end
```

```
x(1:len,10) = M(1:len,1) .* M(1:len,2);
x(1:len,11) = M(1:len,2) .* M(1:len,3);
x(1:len,12) = M(1:len,3) .* M(1:len,4);
x(1:len,13) = M(1:len,1) .* M(1:len,3);
x(1:len,14) = M(1:len,2) .* M(1:len,4);
x(1:len,15) = M(1:len,1) .* M(1:len,4);

T = M(1:len,A+1);

w = pinv(x)*T;
W = w;
```

\# \# \#